CMPSC265 Data Structures and Algorithms
Homework 9

Due: Tuesday, Nov 12rd, 2019, 11:59pm

Learning Objectives

- Binary heap
- Hash table implementation and application

Deliverables:

- The modified java file MinHeap.java for Problem 1.
- *The modified java file Min_PriorityQueue.java* for Problem 1.
- The modified java file HashTable_Quadratic.java for Problem 2.
- Your Java file DogsAndChihuahua.java for problem 3
- Your Java file Anagrams.java for problem 4

Instructions:

- Please first download the Java source file MinHeap.java, Min_PriorityQueue.java and Hashtable_Quadratic.java, modify the codes based upon requirements, and submit the updated source file. Please do NOT change the provided part of the codes, but you may add additional methods if needed.
- Please add the required *credit comments* and *comments* to each Java program you submit.
- **Please TRY YOUR BEST before seeking help from any online resources or tutors or other people. If you have asked for any kind of help, you need also submit your own draft before getting the help, and write a report of how you got helped and what you have learned from the help.**

Problem Specifications:

Problem 1: Implementing Minimum Priority Queue using Min_Heap (25')

The attached MinHeap.java contains the definition of the MinHeap class.

The attached Min_PrioriyQueue.java file contains the definition of the Min_PriorityQueue class.

Please finish the implementation of the method of *insert(), trickleUp(), delete()* and *trickleDown()* in the MinHeap class, so that a Min_Heap can be constructed and maintained.

Make sure a Min_Heap is a complete binary tree data structure that satisfies the heap ordering property, so that each node's key will be less than (or equal to) the key of its two children, and therefore, the root node contains the minimum key.

Each time when deleting a node from the MinHeap, we always delete the root node that contains the minimum key. This shows that MinHeap can provide a good implementation of the Minimum Priority Queue. Please finish the implementation of the Min_PriorityQueue class as required.

Outputs:

```
$ java Min_PriorityQueue
10 20 30 40 50
```

Problem 2: ***HashTable of strings with Quadratic Probing (25')***

Description:

Please finish definition of the HashTable_Quadratic class, so that a hash table for strings can be built, and using **quadratic probing** rather than linear-probing to solve collision.

To compute for the hash value of a given string, for example, CLRS, you can use the following approach:

1) First, We need to first convert the string to be an integer. One way to do this is to use the ASCII value of every character.
   - ASCII values: C=67, L=76, R=82, S=83.
   - There are 128 basic ASCII values.
   - So, CLRS = $67 \cdot 128^3 + 76 \cdot 128^2 + 82 \cdot 128^1 + 83 \cdot 128^0$ = 141,764,947.

2) Secondly, transform the integer into an index in the array representing the hash table. Suppose the size of the hash table is M,
    - The hash value of 141,764,947 would be:
    - 141,764,947 % M

Make sure you are using Quadratic probing to solve collision.

Outputs:

```
$java HashTable_Quadratic
Enter size of hash table: 11
Table: horse apple jungle car ice dog giraff book ** egg
fish
```

Problem 3: Chihuahua and Dogs (25')

Description:

You're given strings J representing the types of dogs that are chihuahua, and S representing the dogs you have. Each character in S is a type of dog you have. You want to know how many of the dogs you have are also Chihuahua.

You may assume that the letters in J are guaranteed distinct, and all characters in J and S are letters. Letters are case sensitive, so "a" is considered a different type of stone from "A".

For example:
Input: J = "aA", S = "aAAbbbb"
Output: 3

Input: J = "z", S = "ZZ"
Output: 0

Please write a Java program to implement this. Make sure you are using HashTable (also this would be best choice).

Outputs:

```
$java DogsAndChihuahua
Please enter a non-empty string representing the types of
CHihuahua: aA
```

```
Please enter a non-empty string representing the types of
dogs: aAAbbbb
There are three Chihuahua.
```

Problem 4: Anagrams (25')

Description:

Given an array of strings, group anagrams together.
Example:
Input: ["eat", "tea", "tan", "ate", "nat", "bat"],
Output:

```
[
  ["ate","eat","tea"],
  ["nat","tan"],
  ["bat"]
]
```

Anagram is a word, phrase, or name formed by rearranging the letters of another, such as cinema, formed from iceman (both contain the letter as 'c', 'i', 'n', 'e', 'm', 'a', but in different order). Therefore, cinema and iceman are anagram of each other.

Please write a Java program using to implement this. Make sure you are using HashTable (actually that would also be the best choice).

Outputs:

$java anagrams

```
Please enter a series of strings separated by a space:
Eat tea tan ata nat bat
There are three groups of anagrams, and they are:
[ate, eat, tea]
[nat, tan]
[bat]
```